US 20230091941A1

(19) **United States**

(12) **Patent Application Publication** (10) **Pub. No.: US 2023/0091941 A1**

Kipnis (43) **Pub. Date: Mar. 23, 2023**

(54) **FLOW CONTROL INTEGRITY**

(71) Applicant: **Mobileye Vision Technologies Ltd.**, Jerusalem (IL)

(72) Inventor: **Aviad Kipnis**, Efrat (IL)

(21) Appl. No.: **17/439,125**

(22) PCT Filed: **Mar. 31, 2021**

(86) PCT No.: **PCT/IB2021/000196**

§ 371 (c)(1),
(2) Date: **Sep. 14, 2021**

Related U.S. Application Data

(60) Provisional application No. 63/003,494, filed on Apr. 1, 2020.

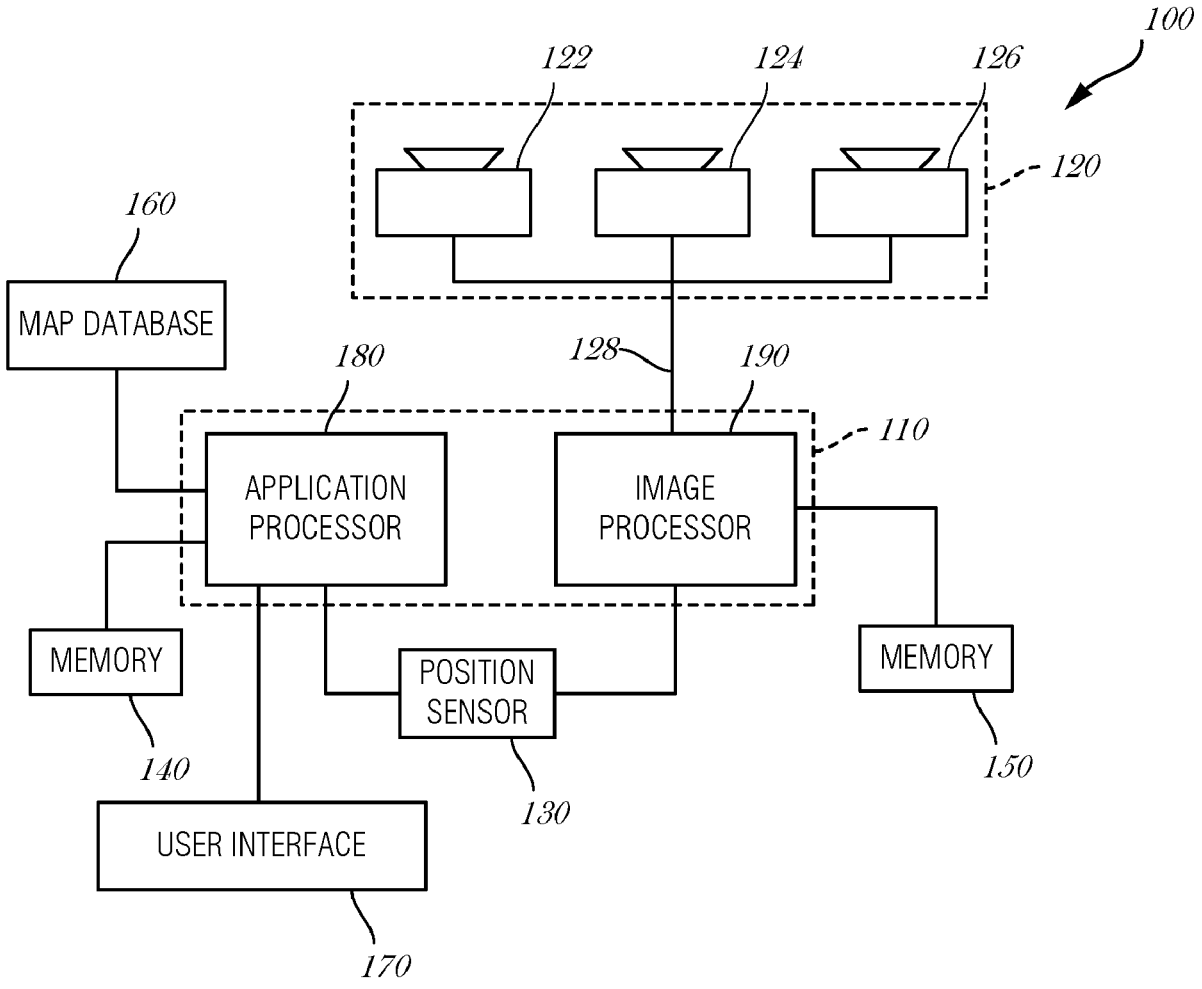
Publication Classification

(51) **Int. Cl.**
G06F 21/52 (2006.01)
G06F 21/55 (2006.01)

(52) **U.S. Cl.**
CPC ***G06F 21/52*** (2013.01); ***G06F 21/554*** (2013.01)

(57) **ABSTRACT**

A method for evaluating flow control integrity, the method may include detecting that a flow reached a flow change command or is about to reach the flow change command, wherein the flow change command belongs to a current software environment, wherein the current software environment is identified by a current environment identifier; retrieving a shadow environment identifier that is a last environment identifier stored in a shadow stack, wherein the shadow environment identifier identifies a software environment having an entry region that was a last entry region accessed by the flow, wherein the entry region comprises a shadow stack update instruction that was executed by the flow; comparing the shadow environment identifier to the current environment identifier; and detecting a potential attack when the shadow environment identifier differs from the current environment identifier.



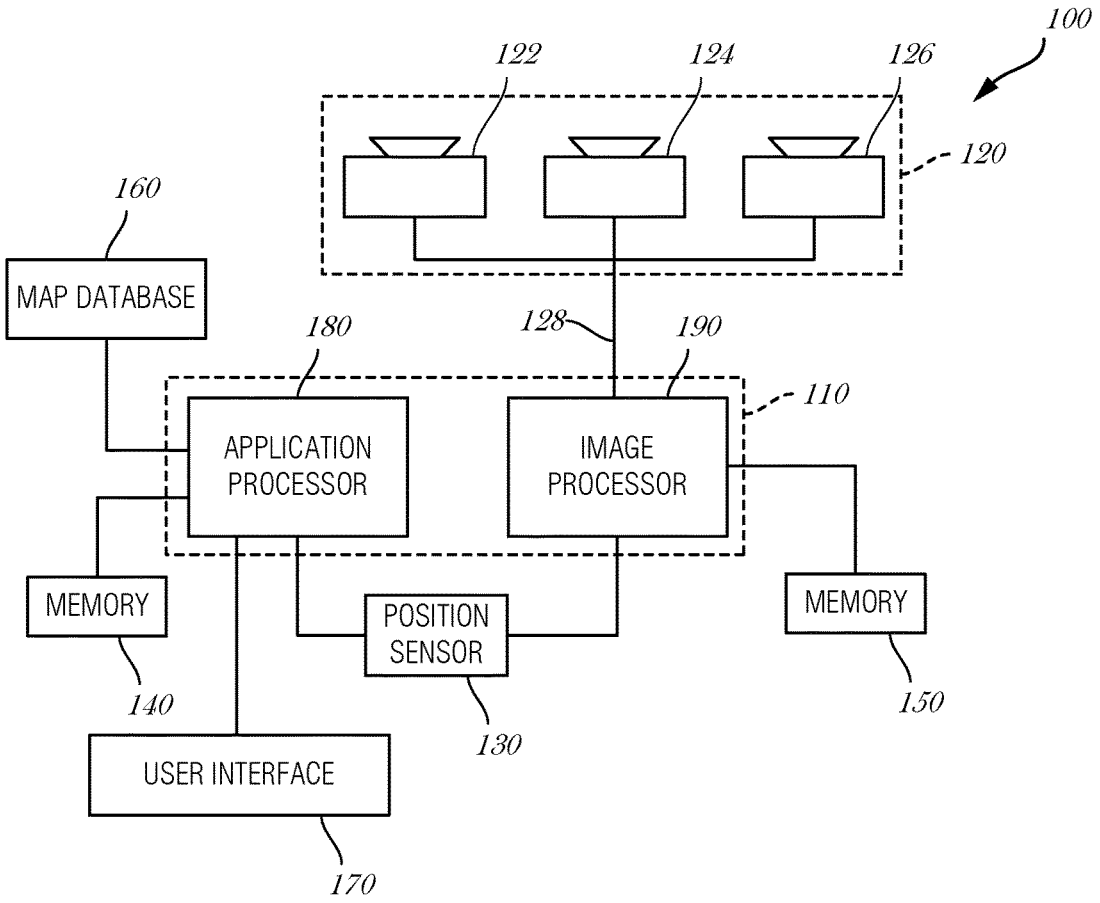

100
122
124
126
120
160
MAP DATABASE
180
128
190
110
APPLICATION PROCESSOR
IMAGE PROCESSOR
MEMORY
POSITION SENSOR
MEMORY
140
130
150
USER INTERFACE
170

FIG. 1

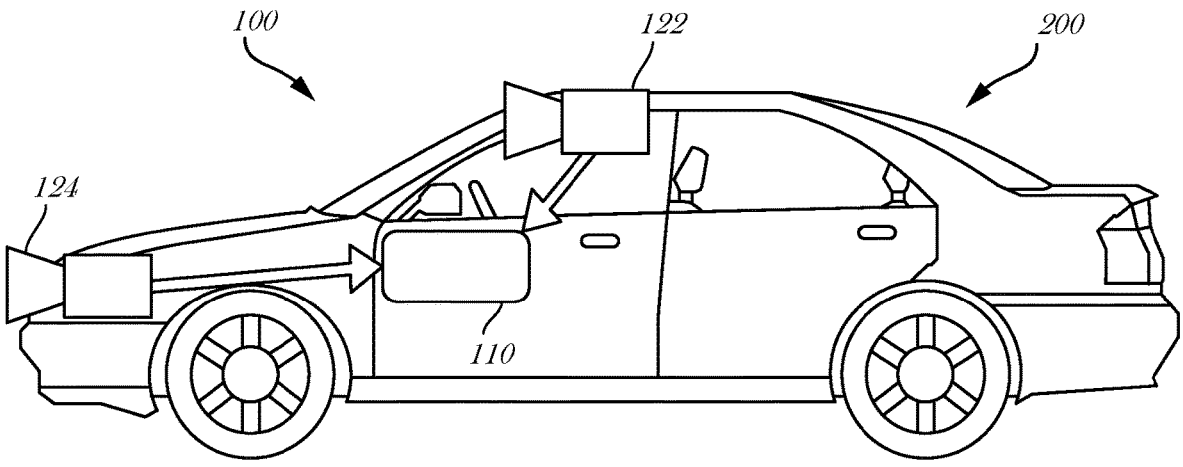

100
122
200
124
110

FIG. 2A

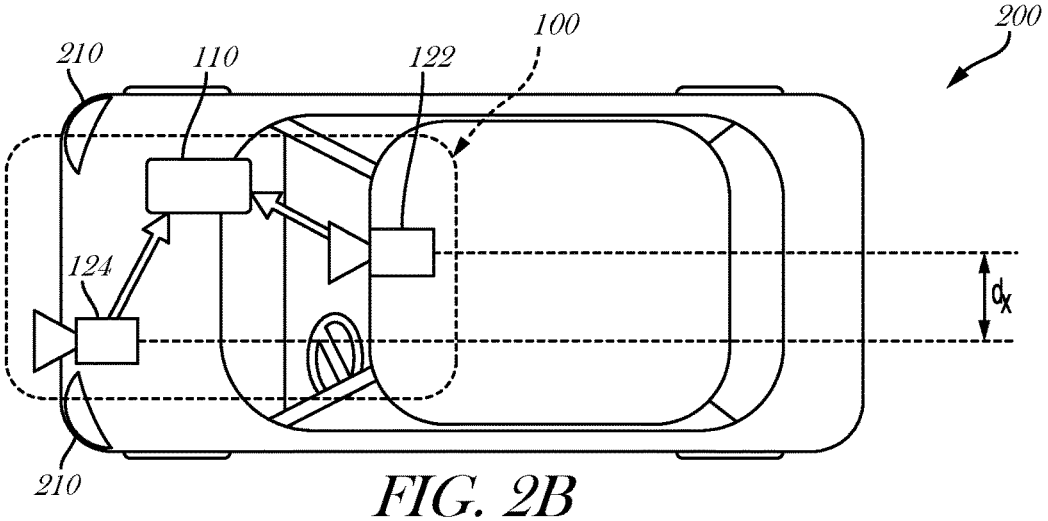


FIG. 2B

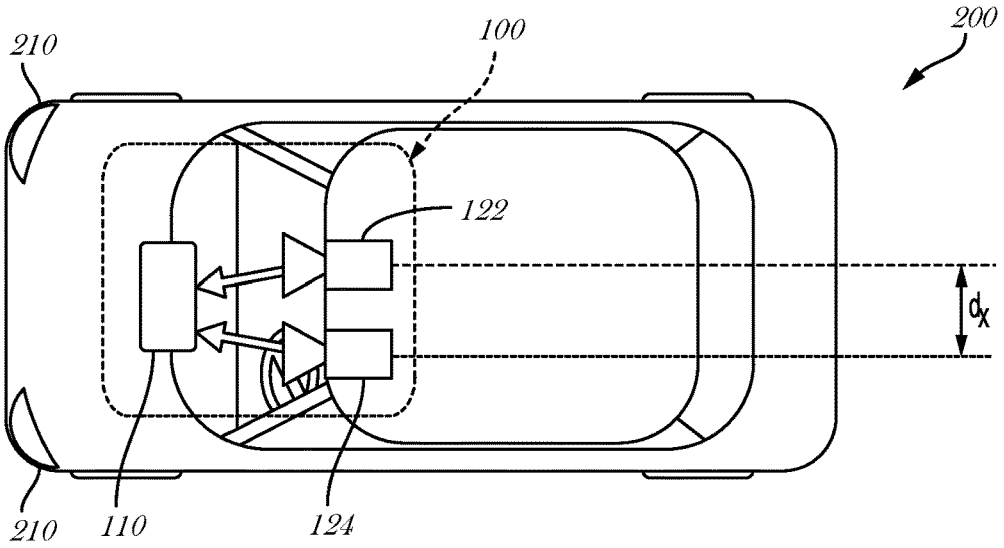


FIG. 2C

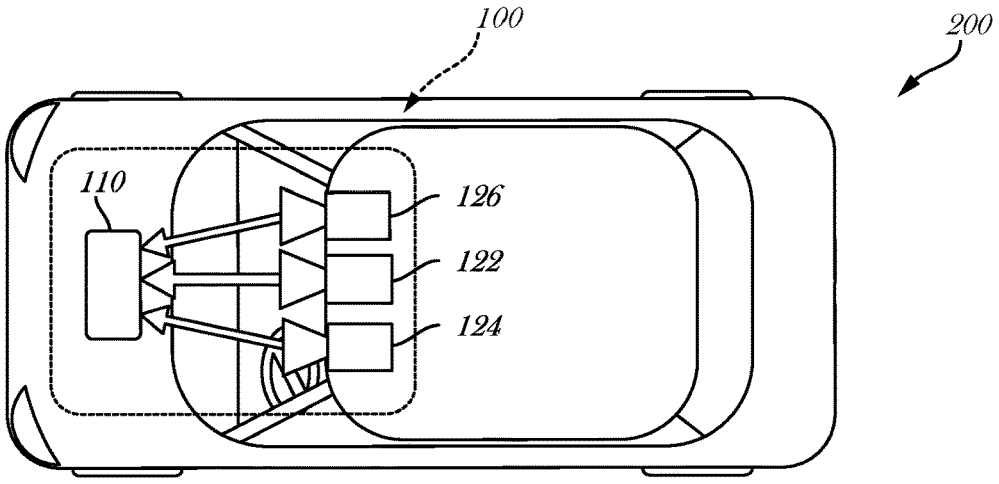


FIG. 2D

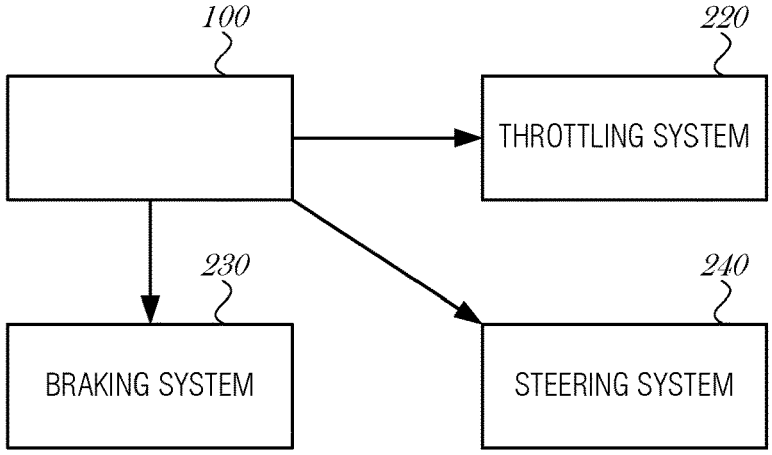
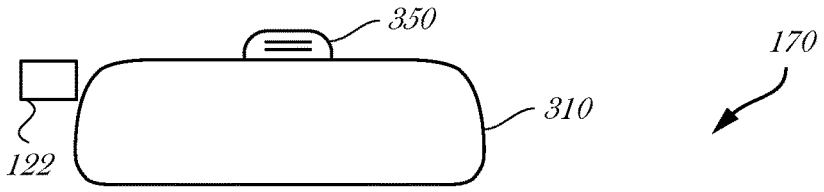

100
220
THROTTLING SYSTEM
230
240
BRAKING SYSTEM
STEERING SYSTEM

FIG. 2E


350
170
310
122

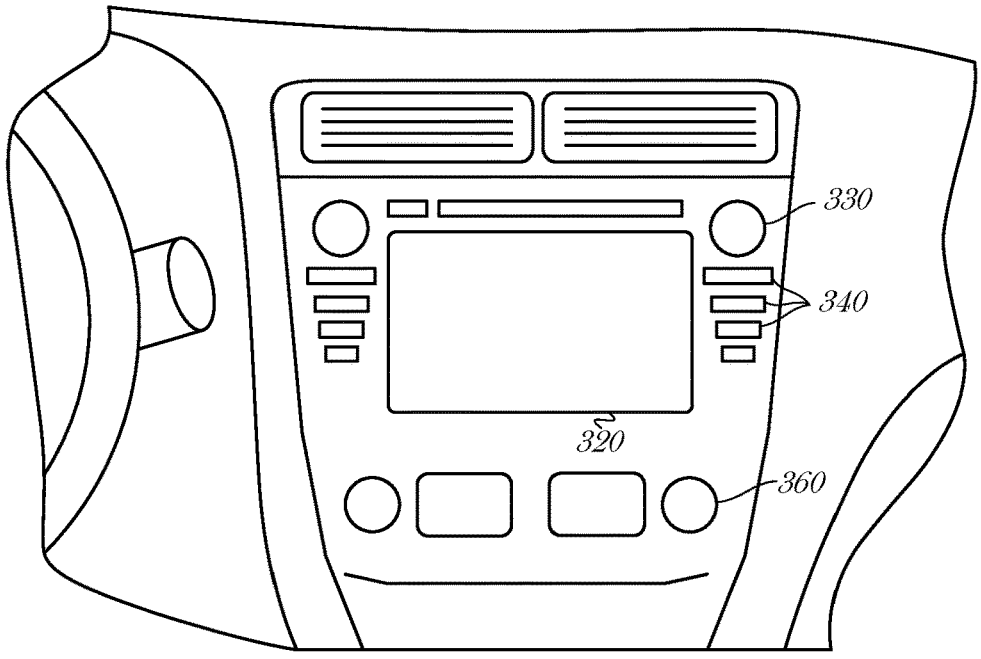

330
340
320
360

FIG. 3

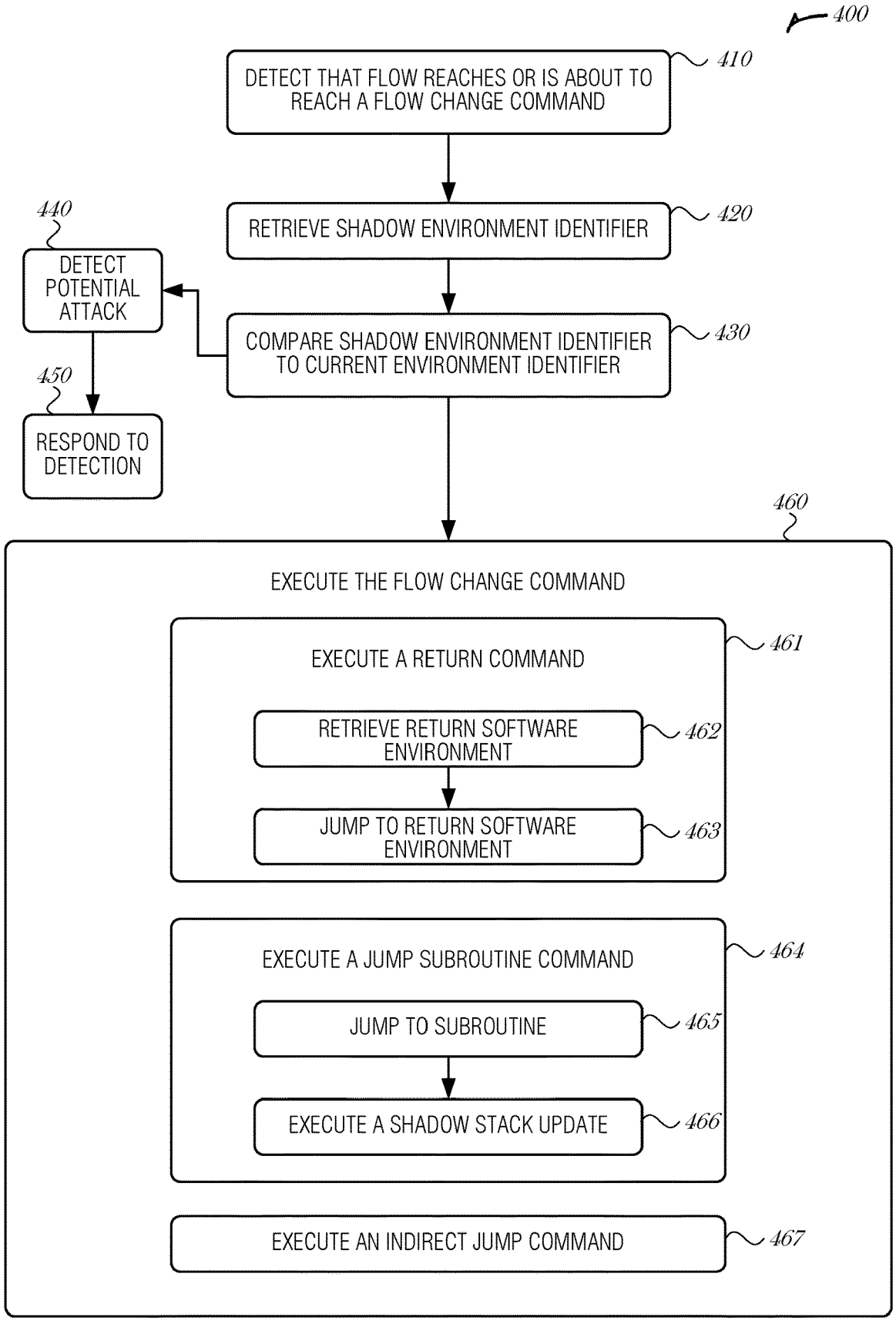

400
DETECT THAT FLOW REACHES OR IS ABOUT TO REACH A FLOW CHANGE COMMAND
410
RETRIEVE SHADOW ENVIRONMENT IDENTIFIER
420
COMPARE SHADOW ENVIRONMENT IDENTIFIER TO CURRENT ENVIRONMENT IDENTIFIER
430
440
DETECT POTENTIAL ATTACK
450
RESPOND TO DETECTION
460
EXECUTE THE FLOW CHANGE COMMAND
EXECUTE A RETURN COMMAND
461
RETRIEVE RETURN SOFTWARE ENVIRONMENT
462
JUMP TO RETURN SOFTWARE ENVIRONMENT
463
EXECUTE A JUMP SUBROUTINE COMMAND
464
JUMP TO SUBROUTINE
465
EXECUTE A SHADOW STACK UPDATE
466
EXECUTE AN INDIRECT JUMP COMMAND
467

FIG. 4

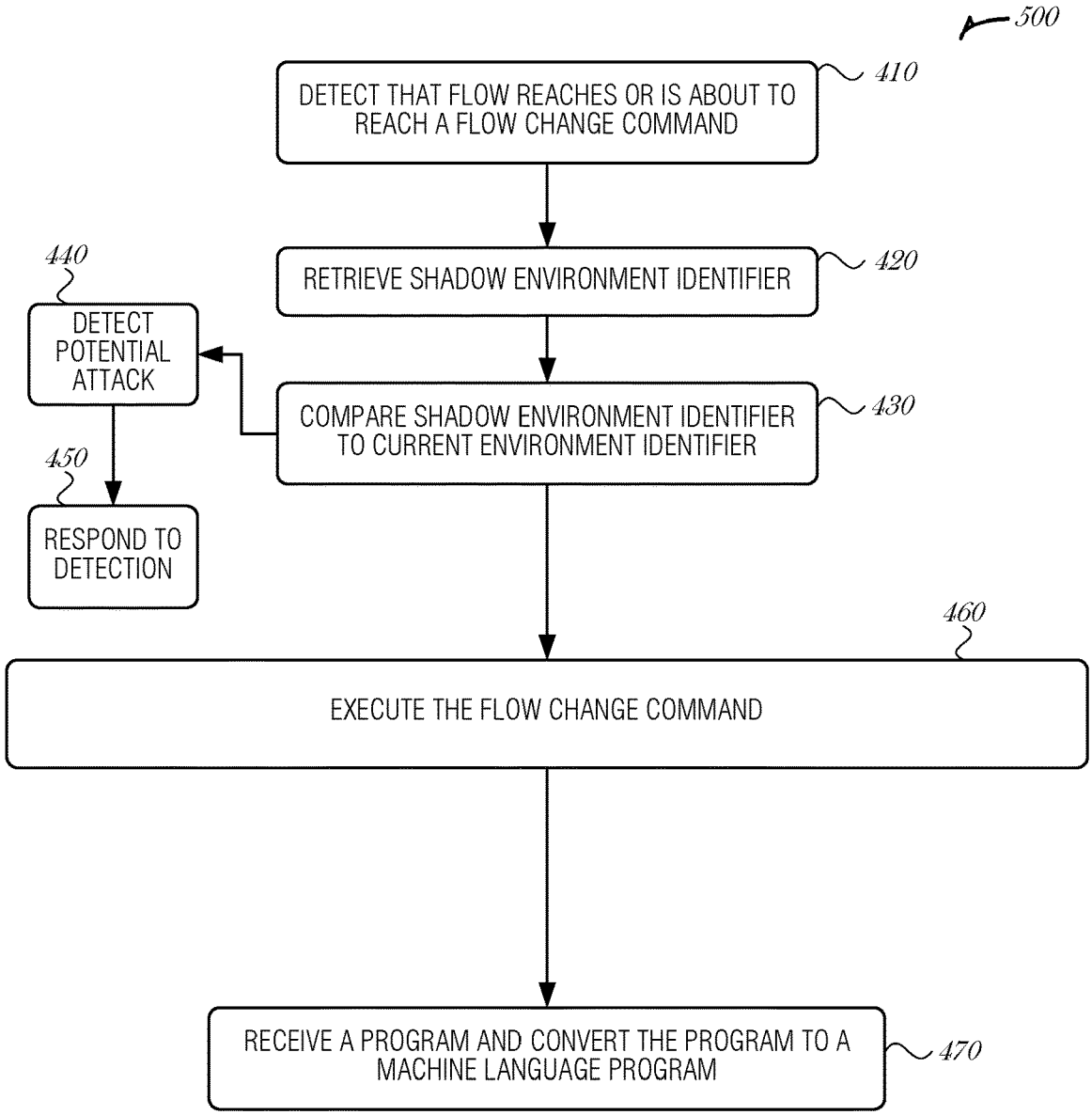

500
DETECT THAT FLOW REACHES OR IS ABOUT TO REACH A FLOW CHANGE COMMAND
410
RETRIEVE SHADOW ENVIRONMENT IDENTIFIER
420
COMPARE SHADOW ENVIRONMENT IDENTIFIER TO CURRENT ENVIRONMENT IDENTIFIER
430
440
DETECT POTENTIAL ATTACK
450
RESPOND TO DETECTION
460
EXECUTE THE FLOW CHANGE COMMAND
RECEIVE A PROGRAM AND CONVERT THE PROGRAM TO A MACHINE LANGUAGE PROGRAM
470

FIG. 5

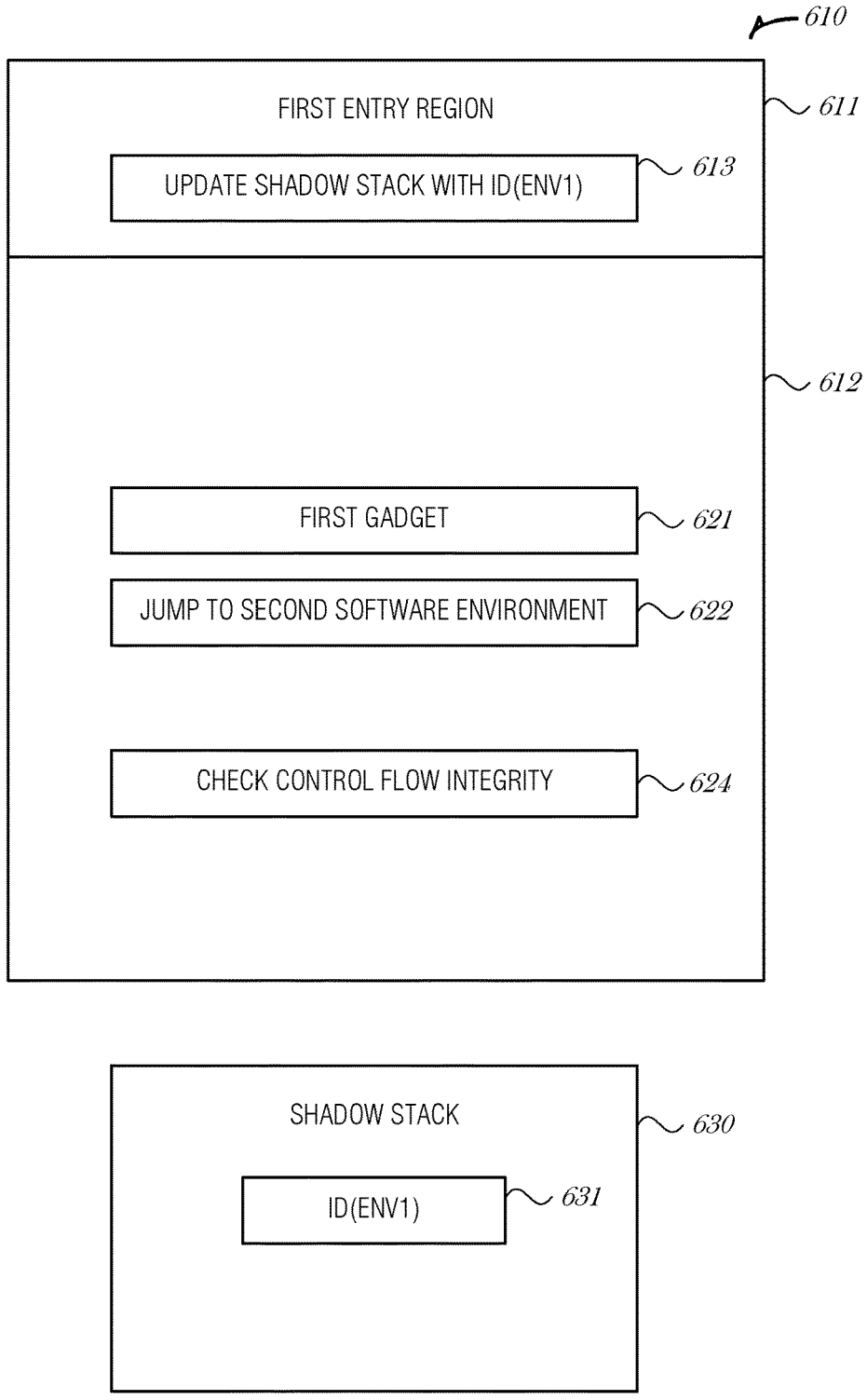

610
FIRST ENTRY REGION
611
UPDATE SHADOW STACK WITH ID(ENV1)
613
612
FIRST GADGET
621
JUMP TO SECOND SOFTWARE ENVIRONMENT
622
CHECK CONTROL FLOW INTEGRITY
624
SHADOW STACK
630
ID(ENV1)
631

FIG. 6

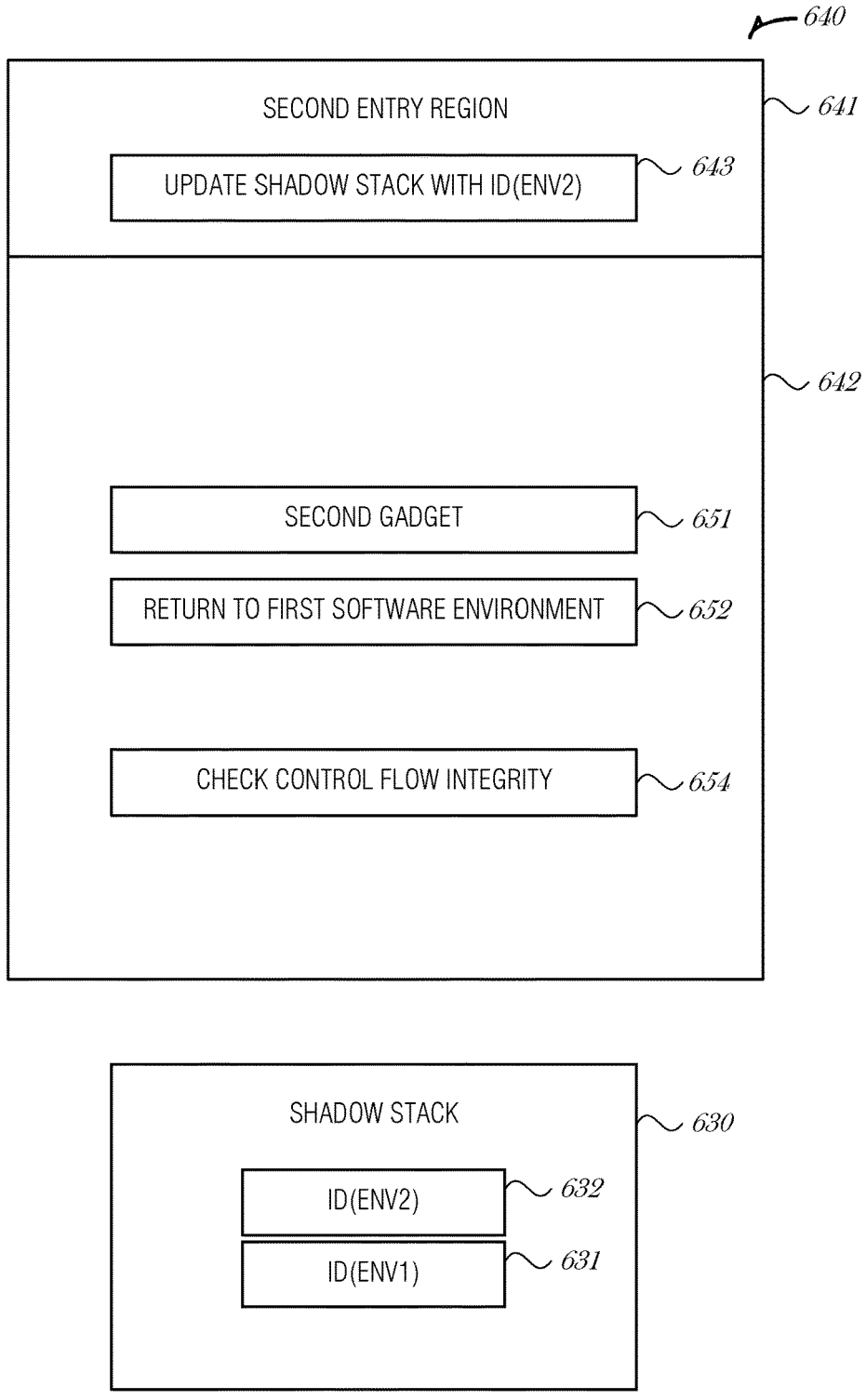

640
SECOND ENTRY REGION
641
UPDATE SHADOW STACK WITH ID(ENV2)
643
642
SECOND GADGET
651
RETURN TO FIRST SOFTWARE ENVIRONMENT
652
CHECK CONTROL FLOW INTEGRITY
654
SHADOW STACK
630
ID(ENV2)
632
ID(ENV1)
631

FIG. 7

FLOW CONTROL INTEGRITY

CLAIM TO PRIORITY

[0001] This patent application claims the benefit of priority to U.S. Provisional Patent Application Ser. No. 63/003,494, filed on Apr. 1, 2020, which is incorporated herein by reference in its entirety.

BACKGROUND

[0002] Advanced driver assistance systems (ADAS), and autonomous vehicle (AV) systems use cameras and other sensors together with object classifiers, which are designed to detect specific objects in an environment of a vehicle navigating a road. Object classifiers are designed to detect predefined objects and are used within ADAS and AV systems to control the vehicle or alert a driver based on the type of object that is detected its location, etc.

[0003] As ADAS and AV systems progress towards fully autonomous operation, it would be beneficial to protect data generated by these systems.

SUMMARY

[0004] The following detailed description refers to the accompanying drawings. Wherever possible, the same reference numbers are used in the drawings and the following description to refer to the same or similar parts. While several illustrative embodiments are described herein, modifications, adaptations and other implementations are possible. For example, substitutions, additions, or modifications may be made to the components illustrated in the drawings, and the illustrative methods described herein may be modified by substituting, reordering, removing, or adding steps to the disclosed methods. Accordingly, the following detailed description is not limited to the disclosed embodiments and examples.

[0005] Disclosed embodiments provide systems and methods that can be used as part of or in combination with autonomous navigation/driving and/or driver assist technology features. Driver assist technology refers to any suitable technology to assist drivers in the navigation and/or control of their vehicles, such as forward collision warning (FCW), lane departure warning (LDW), and traffic sign recognition (TSR), as opposed to fully autonomous driving. In various embodiments, the system may include one, two or more cameras mountable in a vehicle and an associated processor that monitor the environment of the vehicle. In further embodiments, additional types of sensors can be mounted in the vehicle ad can be used in the autonomous navigation and/or driver assist system. In some examples of the presently disclosed subject matter, the system may provide techniques for processing images of an environment ahead of a vehicle navigating a road for training a neural networks or deep learning algorithms to estimate a future path of a vehicle based on images. In yet further examples of the presently disclosed subject matter, the system may provide techniques for processing images of an environment ahead of a vehicle navigating a road using a trained neural network to estimate a future path of the vehicle.

[0006] There are provided systems, methods, as illustrated in the claims and the specification.

[0007] Any combination of any subject matter of any claim may be provided.

[0008] Any combination of any method and/or method step disclosed in any figure and/or in the specification may be provided.

[0009] Any combination of any unit, device, and/or component disclosed in any figure and/or in the specification may be provided. Non-limiting examples of such units include a gather unit, an image processor and the like.

BRIEF DESCRIPTION OF THE DRAWINGS

[0010] The subject matter regarded as the invention is particularly pointed out and distinctly claimed in the concluding portion of the specification. The invention, however, both as to organization and method of operation, together with objects, features, and advantages thereof, may best be understood by reference to the following detailed description when read with the accompanying drawings in which:

[0011] FIG. **1** is a block diagram representation of a system consistent with the disclosed embodiments;

[0012] FIG. **2**A is a diagrammatic side view representation of an exemplary vehicle including a system consistent with the disclosed embodiments;

[0013] FIG. **2**B is a diagrammatic top view representation of the vehicle and system shown in FIG. **2**A consistent with the disclosed embodiments;

[0014] FIG. **2**C is a diagrammatic top view representation of another embodiment of a vehicle including a system consistent with the disclosed embodiments;

[0015] FIG. **2**D is a diagrammatic top view representation of yet another embodiment of a vehicle including a system consistent with the disclosed embodiments;

[0016] FIG. **2**E is a diagrammatic representation of exemplary vehicle control systems consistent with the disclosed embodiments;

[0017] FIG. **3** is a diagrammatic representation of an interior of a vehicle including a rearview mirror and a user interface for a vehicle imaging system consistent with the disclosed embodiments;

[0018] FIG. **4** illustrates an example of a method;

[0019] FIG. **5** illustrates another example of a method;

[0020] FIG. **6** illustrates an example of a first software environment and a shadow stack; and

[0021] FIG. **7** illustrates an example of a second software environment and a shadow stack.

DETAILED DESCRIPTION OF THE DRAWINGS

[0022] In the following detailed description, numerous specific details are set forth in order to provide a thorough understanding of the invention. However, it will be understood by those skilled in the art that the present invention may be practiced without these specific details. In other instances, well-known methods, procedures, and components have not been described in detail so as not to obscure the present invention.

[0023] The subject matter regarded as the invention is particularly pointed out and distinctly claimed in the concluding portion of the specification. The invention, however, both as to organization and method of operation, together with objects, features, and advantages thereof, may best be understood by reference to the following detailed description when read with the accompanying drawings.

[0024] It will be appreciated that for simplicity and clarity of illustration, elements shown in the figures have not necessarily been drawn to scale. For example, the dimen-

sions of some of the elements may be exaggerated relative to other elements for clarity. Further, where considered appropriate, reference numerals may be repeated among the figures to indicate corresponding or analogous elements.

[0025] Because the illustrated embodiments of the present invention may for the most part, be implemented using electronic components and circuits known to those skilled in the art, details will not be explained in any greater extent than that considered necessary as illustrated above, for the understanding and appreciation of the underlying concepts of the present invention and in order not to obfuscate or distract from the teachings of the present invention.

[0026] Any reference in the specification to a method should be applied mutatis mutandis to a system capable of executing the method and should be applied mutatis mutandis to a non-transitory computer readable medium that stores instructions that once executed by a computer result in the execution of the method.

[0027] Any reference in the specification to a system and any other component should be applied mutatis mutandis to a method that may be executed by the memory device and should be applied mutatis mutandis to a non-transitory computer readable medium that stores instructions that may be executed by the memory device. For example, there may be provided a method and/or method steps executed by the image processor described in any one of claims. For example, there may be provided a method and/or method steps executed by the image processor described in any one of claims.

[0028] Any reference in the specification to a non-transitory computer readable medium should be applied mutatis mutandis to a system capable of executing the instructions stored in the non-transitory computer readable medium and should be applied mutatis mutandis to method that may be executed by a computer that reads the instructions stored in the non-transitory computer readable medium.

[0029] Any combination of any module or unit listed in any of the figures, any part of the specification and/or any claims may be provided. Especially any combination of any claimed feature may be provided.

[0030] A pixel may be a picture element obtained by a camera or may be a processed picture element.

[0031] Before discussing in detail examples of features of the processing images of an environment ahead of a vehicle navigating a road for training a neural networks or deep learning algorithms to estimate a future path of a vehicle based on images or feature of the processing of images of an environment ahead of a vehicle navigating a road using a trained neural network to estimate a future path of the vehicle, there is provided a description of various possible implementations and configurations of a vehicle mountable system that can be used for carrying out and implementing the methods according to examples of the presently disclosed subject matter. In some embodiments, various examples of the system can be mounted in a vehicle, and can be operated while the vehicle is in motion. In some embodiments, the system can implement the methods according to examples of the presently disclosed subject matter.

[0032] However, it would be appreciated that embodiments of the present disclosure are not limited to scenarios where a suspected upright object indication is caused by a high-grade road. The suspected upright object indication can be associated with various other circumstances, and can result from other types of image data and also from data that is not image-based or is not exclusively image-based, as well.

[0033] FIG. **1**, to which reference is now made, is a block diagram representation of a system consistent with the disclosed embodiments. System **100** can include various components depending on the requirements of a particular implementation. In some examples, system **100** can include a processing unit **110**, an image acquisition unit **120** and one or more memory units **140**, **150**. Processing unit **110** can include one or more processing devices. In some embodiments, processing unit **110** can include an application processor **180**, an image processor **190**, or any other suitable processing device. Similarly, image acquisition unit **120** can include any number of image acquisition units and components depending on the requirements of a particular application. In some embodiments, image acquisition unit **120** can include one or more image capture devices (e.g., cameras), such as image capture device **122**, image capture device **124**, and image capture device **126**. In some embodiments, system **100** can also include a data interface **128** communicatively connecting processing unit **110** to image acquisition unit **120**. For example, data interface **128** can include any wired and/or wireless link or links for transmitting image data acquired by image acquisition unit **120** to processing unit **110**.

[0034] Both application processor **180** and image processor **190** can include various types of processing devices. For example, either or both of application processor **180** and image processor **190** can include one or more microprocessors, preprocessors (such as image preprocessors), graphics processors, central processing units (CPUs), support circuits, digital signal processors, integrated circuits, memory, or any other types of devices suitable for running applications and for image processing and analysis. In some embodiments, application processor **180** and/or image processor **190** can include any type of single or multi-core processor, mobile device microcontroller, central processing unit, etc. Various processing devices can be used, including, for example, processors available from manufacturers such as Intel®, AMD®, etc. and can include various architectures (e.g., x86 processor, ARM®, etc.).

[0035] In some embodiments, application processor **180** and/or image processor **190** can include any of the EyeQ series of processor chips available from Mobileye®. These processor designs each include multiple processing units with local memory and instruction sets. Such processors may include video inputs for receiving image data from multiple image sensors and may also include video out capabilities. In one example, the EyeQ2® uses 90 nm-micron technology operating at 332 Mhz. The EyeQ2® architecture has two floating point, hyper-thread 32-bit RISC CPUs (MIPS32® 34K® cores), five Vision Computing Engines (VCE), three Vector Microcode Processors (VMP®), Denali 64-bit Mobile DDR Controller, 128-bit internal Sonics Interconnect, dual 16-bit Video input and 18-bit Video output controllers, 16 channels DMA and several peripherals. The MIPS34K CPU manages the five VCEs, three VMP™ and the DMA, the second MIPS34K CPU and the multi-channel DMA as well as the other peripherals. The five VCEs, three VMP® and the MIPS34K CPU can perform intensive vision computations required by multi-function bundle applications. In another example, the EyeQ3®, which is a third-generation processor and is six

times more powerful that the EyeQ2®, may be used in the disclosed examples. In yet another example, the EyeQ4®, the fourth-generation processor, may be used in the disclosed examples.

[0036] While FIG. **1** depicts two separate processing devices included in processing unit **110**, more or fewer processing devices can be used. For example, in some examples, a single processing device may be used to accomplish the tasks of application processor **180** and image processor **190**. In other embodiments, these tasks can be performed by more than two processing devices.

[0037] Processing unit **110** can include various types of devices. For example, processing unit **110** may include various devices, such as a controller, an image preprocessor, a central processing unit (CPU), support circuits, digital signal processors, integrated circuits, memory, or any other types of devices for image processing and analysis. The image preprocessor can include a video processor for capturing, digitizing, and processing the imagery from the image sensors. The CPU can include any number of microcontrollers or microprocessors. The support circuits can be any number of circuits generally well known in the art, including cache, power supply, clock, and input-output circuits. The memory can store software that, when executed by the processor, controls the operation of the system. The memory can include databases and image processing software, including a trained system, such as a neural network, for example. The memory can include any number of random access memories, read only memories, flash memories, disk drives, optical storage, removable storage, and other types of storage. In one instance, the memory can be separate from the processing unit **110**. In another instance, the memory can be integrated into the processing unit **110**.

[0038] Each memory **140**, **150** can include software instructions that when executed by a processor (e.g., application processor **180** and/or image processor **190**), can control operation of various aspects of system **100**. These memory units can include various databases and image processing software. The memory units can include random access memory, read only memory, flash memory, disk drives, optical storage, tape storage, removable storage, and/or any other types of storage. In some examples, memory units **140**, **150** can be separate from the application processor **180** and/or image processor **190**. In other embodiments, these memory units can be integrated into application processor **180** and/or image processor **190**.

[0039] In some embodiments, the system can include a position sensor **130**. The position sensor **130** can include any type of device suitable for determining a location associated with at least one component of system **100**. In some embodiments, position sensor **130** can include a GPS receiver. Such receivers can determine a user position and velocity by processing signals broadcasted by global positioning system satellites. Position information from position sensor **130** can be made available to application processor **180** and/or image processor **190**.

[0040] In some embodiments, the system **100** can be operatively connectible to various systems, devices, and units onboard a vehicle in which the system **100** can be mounted, and through any suitable interfaces (e.g., a communication bus) the system **100** can communicate with the vehicle's systems. Examples of vehicle systems with which the system **100** can cooperate include: a throttling system, a braking system, and a steering system.

[0041] In some embodiments, the system **100** can include a user interface **170**. User interface **170** can include any device suitable for providing information to or for receiving inputs from one or more users of system **100**, including, for example, a touchscreen, microphone, keyboard, pointer devices, track wheels, cameras, knobs, buttons, etc. Information can be provided by the system **100**, through the user interface **170**, to the user.

[0042] In some embodiments, the system **100** can include a map database **160**. The map database **160** can include any type of database for storing digital map data. In some examples, map database **160** can include data relating to a position, in a reference coordinate system, of various items, including roads, water features, geographic features, points of interest, etc. Map database **160** can store not only the locations of such items, but also descriptors relating to those items, including, for example, names associated with any of the stored features and other information about them. For example, locations and types of known obstacles can be included in the database, information about a topography of a road or a grade of certain points along a road, etc. In some embodiments, map database **160** can be physically located with other components of system **100**. Alternatively, or additionally, map database **160** or a portion thereof can be located remotely with respect to other components of system **100** (e.g., processing unit **110**). In such embodiments, information from map database **160** can be downloaded over a wired or wireless data connection to a network (e.g., over a cellular network and/or the Internet, etc.).

[0043] Image capture devices **122**, **124**, and **126** can each include any type of device suitable for capturing at least one image from an environment. Moreover, any number of image capture devices can be used to acquire images for input to the image processor. Some examples of the presently disclosed subject matter can include or can be implemented with only a single-image capture device, while other examples can include or can be implemented with two, three, or even four or more image capture devices. Image capture devices **122**, **124**, and **126** will be further described with reference to FIGS. **2**A-**2**E, below.

[0044] It would be appreciated that the system **100** can include or can be operatively associated with other types of sensors, including for example: an acoustic sensor, a RF sensor (e.g., radar transceiver), a LIDAR sensor. Such sensors can be used independently of or in cooperation with the image acquisition unit **120**. For example, the data from the radar system (not shown) can be used for validating the processed information that is received from processing images acquired by the image acquisition unit **120**, e.g., to filter certain false positives resulting from processing images acquired by the image acquisition unit **120**, or it can be combined with or otherwise compliment the image data from the image acquisition unit **120**, or some processed variation or derivative of the image data from the image acquisition unit **120**.

[0045] System **100**, or various components thereof, can be incorporated into various different platforms. In some embodiments, system **100** may be included on a vehicle **200**, as shown in FIG. **2**A. For example, vehicle **200** can be equipped with a processing unit **110** and any of the other components of system **100**, as described above relative to FIG. **1**. While in some embodiments vehicle **200** can be equipped with only a single-image capture device (e.g., camera), in other embodiments, such as those discussed in

connection with FIGS. **2**A-**2**E, multiple image capture devices can be used. For example, either of image capture devices **122** and **124** of vehicle **200**, as shown in FIG. **2**A, can be part of an ADAS (Advanced Driver Assistance Systems) imaging set.

[0046] The image capture devices included on vehicle **200** as part of the image acquisition unit **120** can be positioned at any suitable location. In some embodiments, as shown in FIGS. **2**A-**2**E and **3**, image capture device **122** can be located in the vicinity of the rearview mirror. This position may provide a line of sight similar to that of the driver of vehicle **200**, which can aid in determining what is and is not visible to the driver.

[0047] Other locations for the image capture devices of image acquisition unit **120** can also be used. For example, image capture device **124** can be located on or in a bumper of vehicle **200**. Such a location can be especially suitable for image capture devices having a wide field of view. The line of sight of bumper-located image capture devices can be different from that of the driver. The image capture devices (e.g., image capture devices **122**, **124**, and **126**) can also be located in other locations. For example, the image capture devices may be located on or in one or both of the side mirrors of vehicle **200**, on the roof of vehicle **200**, on the hood of vehicle **200**, on the trunk of vehicle **200**, on the sides of vehicle **200**, mounted on, positioned behind, or positioned in front of any of the windows of vehicle **200**, and mounted in or near light figures on the front and/or back of vehicle **200**, etc. The image acquisition unit **120**, or an image capture device that is one of a plurality of image capture devices that are used in an image acquisition unit **120**, can have a field-of-view (FOV) that is different than the FOV of a driver of a vehicle, and not always see the same objects. In one example, the FOV of the image acquisition unit **120** can extend beyond the FOV of a typical driver and can thus image objects which are outside the FOV of the driver. In yet another example, the FOV of the image acquisition unit **120** is some portion of the FOV of the driver. In some embodiments, the FOV of the image acquisition unit **120** corresponding to a sector which covers an area of a road ahead of a vehicle and possibly also surroundings of the road.

[0048] In addition to image capture devices, vehicle **200** can be include various other components of system **100**. For example, processing unit **110** may be included on vehicle **200** either integrated with or separate from an engine control unit (ECU) of the vehicle. Vehicle **200** may also be equipped with a position sensor **130**, such as a GPS receiver and may also include a map database **160** and memory units **140** and **150**.

[0049] FIG. **2**A is a diagrammatic side view representation of a vehicle imaging system according to examples of the presently disclosed subject matter. FIG. **2**B is a diagrammatic top view illustration of the example shown in FIG. **2**A. As illustrated in FIG. **2**B, the disclosed examples can include a vehicle **200** including in its body a system **100** with a first image capture device **122** positioned in the vicinity of the rearview mirror and/or near the driver of vehicle **200**, a second image capture device **124** positioned on or in a bumper region (e.g., one of bumper regions **210**) of vehicle **200**, and a processing unit **110**.

[0050] As illustrated in FIG. **2**C, image capture devices **122** and **124** may both be positioned in the vicinity of the rearview mirror and/or near the driver of vehicle **200**. Additionally, while two image capture devices **122** and **124** are shown in FIGS. **2**B and **2**C, it should be understood that other embodiments may include more than two image capture devices. For example, in the embodiment shown in FIG. **2**D, first, second, and third image capture devices **122**, **124**, and **126**, are included in the system **100** of vehicle **200**.

[0051] As shown in FIG. **2**D, image capture devices **122**, **124**, and **126** may be positioned in the vicinity of the rearview mirror and/or near the driver seat of vehicle **200**. The disclosed examples are not limited to any particular number and configuration of the image capture devices, and the image capture devices may be positioned in any appropriate location within and/or on vehicle **200**.

[0052] It is also to be understood that disclosed embodiments are not limited to a particular type of vehicle **200** and may be applicable to all types of vehicles including automobiles, trucks, trailers, motorcycles, bicycles, self-balancing transport devices and other types of vehicles.

[0053] The first image capture device **122** can include any suitable type of image capture device. Image capture device **122** can include an optical axis. In one instance, the image capture device **122** can include an Aptina M9V024 WVGA sensor with a global shutter. In another example, a rolling shutter sensor can be used. Image acquisition unit **120**, and any image capture device which is implemented as part of the image acquisition unit **120**, can have any desired image resolution. For example, image capture device **122** can provide a resolution of 1280×960 pixels and can include a rolling shutter.

[0054] Image acquisition unit **120**, and any image capture device which is implemented as part of the image acquisition unit **120**, can include various optical elements. In some embodiments one or more lenses can be included, for example, to provide a desired focal length and field of view for the image acquisition unit **120**, and for any image capture device which is implemented as part of the image acquisition unit **120**. In some examples, an image capture device which is implemented as part of the image acquisition unit **120** can include or be associated with any optical elements, such as a 6 mm lens or a 12 mm lens, for example. In some examples, image capture device **122** can be configured to capture images having a desired (and known) field-of-view (FOV).

[0055] The first image capture device **122** may have a scan rate associated with acquisition of each of the first series of image scan lines. The scan rate may refer to a rate at which an image sensor can acquire image data associated with each pixel included in a particular scan line.

[0056] FIG. **2**E is a diagrammatic representation of vehicle control systems, according to examples of the presently disclosed subject matter. As indicated in FIG. **2**E, vehicle **200** can include throttling system **220**, braking system **230**, and steering system **240**. System **100** can provide inputs (e.g., control signals) to one or more of throttling system **220**, braking system **230**, and steering system **240** over one or more data links (e.g., any wired and/or wireless link or links for transmitting data). For example, based on analysis of images acquired by image capture devices **122**, **124**, and/or **126**, system **100** can provide control signals to one or more of throttling system **220**, braking system **230**, and steering system **240** to navigate vehicle **200** (e.g., by causing an acceleration, a turn, a lane shift, etc.). Further, system **100** can receive inputs from one or more of throttling system **220**, braking system **230**,

and steering system **240** indicating operating conditions of vehicle **200** (e.g., speed, whether vehicle **200** is braking and/or turning, etc.).

[0057] As shown in FIG. **3**, the vehicle may also include a user interface **170** for interacting with a driver or a passenger of vehicle. For example, user interface **170** in a vehicle application may include a touch screen **320**, knobs **330**, buttons **340**, and a microphone **350**. A driver or passenger of the vehicle may also use handles (e.g., located on or near the steering column of the vehicle including, for example, turn signal handles), buttons (e.g., located on the steering wheel of the vehicle), and the like, to interact with system **100**. In some embodiments, microphone **350** may be positioned adjacent to a rearview mirror **310**. Similarly, in some embodiments, image capture device **122** may be located near rearview mirror **310**. In some embodiments, user interface **170** may also include one or more speakers **360** (e.g., speakers of a vehicle audio system). For example, system **100** may provide various notifications (e.g., alerts) via speakers **360**.

[0058] As will be appreciated by a person skilled in the art having the benefit of this disclosure, numerous variations and/or modifications may be made to the foregoing disclosed embodiments. For example, not all components are essential for the operation of system **100**. Further, any component may be located in any appropriate part of system **100** and the components may be rearranged into a variety of configurations while providing the functionality of the disclosed embodiments. Therefore, the foregoing configurations are examples and, regardless of the configurations discussed above, system **100** can provide a wide range of functionality to analyze the surroundings of the vehicle and, in response to this analysis, navigate and/or otherwise control and/or operate the vehicle. Navigation, control, and/or operation of the vehicle may include enabling and/or disabling (directly or via intermediary controllers, such as the controllers mentioned above) various features, components, devices, modes, systems, and/or subsystems associated with vehicle **200**. Navigation, control, and/or operation may alternately or additionally include interaction with a user, driver, passenger, passerby, and/or other vehicle or user, which may be located inside or outside the vehicle, for example by providing visual, audio, haptic, and/or other sensory alerts and/or indications.

[0059] As discussed below in further detail and consistent with various disclosed embodiments, system **100** may provide a variety of features related to autonomous driving, semi-autonomous driving and/or driver assist technology. For example, system **100** may analyze image data, position data (e.g., GPS location information), map data, speed data, and/or data from sensors included in the vehicle. System **100** may collect the data for analysis from, for example, image acquisition unit **120**, position sensor **130**, and other sensors. Further, system **100** may analyze the collected data to determine whether or not the vehicle should take a certain action, and then automatically take the determined action without human intervention. It would be appreciated that in some cases, the actions taken automatically by the vehicle are under human supervision, and the ability of the human to intervene adjust abort or override the machine action is enabled under certain circumstances or at all times. For example, when vehicle **200** navigates without human intervention, system **100** may automatically control the braking, acceleration, and/or steering of the vehicle (e.g., by sending control signals to one or more of throttling system **220**, braking system **230**, and steering system **240**). Further, system **100** may analyze the collected data and issue warnings, indications, recommendations, alerts, or instructions to a driver, passenger, user, or other person inside or outside of the vehicle (or to other vehicles) based on the analysis of the collected data. Additional details regarding the various embodiments that are provided by system **100** are provided below.

[0060] Return-oriented programming (ROP) (also referred to as Jump oriented programming (JOP)) is a computer security exploit technique that allows an attacker to execute code in the presence of security defenses such as executable space protection and code signing.

[0061] In this technique, an attacker gains control of the call stack (hereinafter "stack") to hijack program control flow and then executes carefully chosen machine instruction sequences that are already present in the machine's memory, called "gadgets". Each gadget typically ends in a return instruction and is located in a subroutine within the existing program and/or shared library code. Chained together, these gadgets allow an attacker to perform arbitrary operations on a machine employing defenses that thwart simpler attacks.

[0062] Return-oriented programming is an advanced version of a stack smashing attack. Generally, these types of attacks arise when an adversary manipulates the call stack by taking advantage of a bug in the program, often a buffer overrun. In a buffer overrun, a function that does not perform proper bounds checking before storing user-provided data into memory will accept more input data than it can store properly. If the data is being written onto the stack, the excess data may overflow the space allocated to the function's variables (e.g., "locals" in the stack) and overwrite the return address. This address will later be used by the function to redirect control flow back to the caller. If it has been overwritten, control flow will be diverted to the location specified by the new return address.

[0063] In a standard buffer overrun attack, the attacker would simply write attack code (the "payload") onto the stack and then overwrite the return address with the location of these newly written instructions. Until the late 1990s, major operating systems did not offer any protection against these attacks. For instance, Microsoft Windows provided no buffer-overrun protections until 2004.

[0064] Eventually, operating systems began to combat the exploitation of buffer overflow bugs by marking the memory where data is written as non-executable, a technique known as executable space protection. With this enabled, the machine would refuse to execute any code located in user-writable areas of memory, preventing the attacker from placing payload on the stack and jumping to it via a return address overwrite. Hardware support later became available to strengthen this protection.

[0065] With data execution prevention, an adversary cannot execute maliciously injected instructions because a typical buffer overflow overwrites contents in the data section of memory, which is marked as non-executable. To defeat this, a return-oriented programming attack does not inject malicious code, but rather uses instructions that are already present, called "gadgets", by manipulating return addresses. A typical data execution prevention cannot defend against this attack because the adversary did not use malicious code but rather combined "good" instructions by

changing return addresses; therefore, the code used would not be marked non-executable.

[0066] Return-oriented programming builds on the borrowed code chunks approach and extends it to provide Turing complete functionality to the attacker, including loops and conditional branches.

[0067] Put another way, return-oriented programming provides a fully functional "language" that an attacker can use to make a compromised machine perform any operation desired. It has been demonstrated how all the important programming constructs can be simulated using return-oriented programming against a target application linked with the C standard library and containing an exploitable buffer overrun vulnerability.

[0068] A return-oriented programming attack is superior to the other attack types discussed both in expressive power and in resistance to defensive measures. None of the counter-exploitation techniques mentioned above, including removing potentially dangerous functions from shared libraries altogether, are effective against a return-oriented programming attack.

[0069] There may be provided a system, a method, and a non-transitory computer readable medium that maintain flow control integrity by detecting return-oriented programming attacks.

[0070] While the return-oriented programming attempts to jump between gadgets following a flow that differs from the original flow of a program, the method verifies that the original (also referred to as proper or non-compromised) flow is maintained.

[0071] When executing instructions of a software environment, the method checks whether the flow passed through the entry region of the software environment and blocks flows that jumped to a gadget without passing through the entry region.

[0072] The entry region may be modified (for example by a compiler) to include a shadow stack update instruction that updates the shadow stack with a shadow environment identifier that identifies the software environment of the entry region.

[0073] When an attempt to change a flow (from a current software environment) is made (or is about to be made), the method checks whether the shadow environment identifier equals a current environment identifier (that identifies the current software environment).

[0074] A mismatch indicates that the flow did not pass through the entry region of the current environment identifier—and a return-oriented programming attack is detected.

[0075] FIG. **4** illustrates an example of method **400** for evaluating flow control integrity. Method **400** may start by step **410** of detecting that a flow reached a flow change command or is about to reach the flow change command. The flow change command belongs to a current software environment. The current software environment is identified by a current environment identifier.

[0076] Examples of a flow change command include a return command (for returning from a direct jump command), an indirect jump command (that may jump to an address within the current software environment), and a jump subroutine command.

[0077] The operation of detecting that a flow is about to reach the flow change command may include searching for the flow change command in a command stack even before the program counter points to the flow change command. The flow change command may be searched within a distance (for example, in proximity—such as between 1 and 10 entries or any other value) from the entry pointed by the program counter.

[0078] Step **410** may be followed by step **420** of retrieving a shadow environment identifier that is a last environment identifier stored in a shadow stack. The last environment identifier may be the most updated environment identifier stored in the shadow stack.

[0079] The shadow environment identifier identifies a software environment having an entry region that was the last entry region that was passed by the flow. The entry region includes a shadow stack update instruction that was executed by the flow.

[0080] Step **420** may be followed by step **430** of comparing the shadow environment identifier to the current environment identifier.

[0081] When the shadow environment identifier differs from the current environment identifier, then step **430** may be followed by step **440** of detecting a potential attack. The potential attack may be a return-oriented programming attack.

[0082] Step **440** may be followed by step **450** of responding to the detecting of the potential attack. The responding may include generating an alert, stopping the execution of the flow, rebooting a processor, and the like.

[0083] When the shadow environment identifier equals the current environment identifier then step **430** may be followed by step **460** of executing the flow change command.

[0084] Step **460** may include at least one of the following steps:

[0085] 1) Step **461** of executing a return command.

[0086] 2) Step **464** of executing a jump subroutine command.

[0087] 3) Step **467** of executing an indirect jump command.

[0088] Step **461** may include steps **462** and **463**.

[0089] Step **462** may include retrieving, from the shadow stack, a return environment identifier that identifies a return software environment. Step **462** may be followed by step **463** of jumping to (returning to) the return software environment.

[0090] Step **464** may include steps **465** and **466**. Step **465** may include jumping to the subroutine. Step **466** may include executing a shadow stack update included in the entry region of the subroutine. Step **466** may include storing in the shadow stack a shadow environment identifier that identifies the subroutine.

[0091] FIG. **5** is another example of method **500**. Method **500** differs from method **400** by including step **470** of receiving a program (for example a high level program) and converting, by a complier, the program to a machine language program (also referred to as code). The converting may include modifying an entry region of at least one software environment of the machine language program to include a shadow stack update instruction that updates the shadow stack with a shadow environment identifier that identifies the software environment that includes the entry region.

[0092] In addition, FIG. **5** does not illustrate (for simplicity of explanation) steps **461-467**.

[0093] FIG. **6** illustrates an example of a first software environment **610** and a shadow stack **630**. The first software environment **610** includes a first entry region **611** and one or

more other parts **612** that include a first gadget **621**, a flow change command such as "jump to second software environment" command **622**, and a check control flow integrity command **624**. The jump command may be a direct or indirect jump command.

[0094] If the flow executed the first software environment **610** in a proper manner, starting from executing instructions of the first entry region **611**, then the flow will execute a shadow stack update instruction such as "update shadow stack with ID(ENV1)" **613**, where ID(ENV1) identifies the first software environment **610**.

[0095] If the flow arrives to the first software environment **610** as result of a return-oriented programming attack, without passing through the first entry region **611**, then the shadow stack will not be updated with ID(ENV1).

[0096] The instructions within the one or more other parts **612** are executed until reaching (or before reaching) the control flow integrity command **624** that will compare ID(ENV1) to a shadow environment identifier stored in the shadow stack **630** and respond according to the outcome of the comparison.

[0097] FIG. **6** illustrates the scenario in which the flow executed the first software environment **610** in a proper manner and executed "update shadow stack with ID(ENV1)" **613**, so that the shadow environment identifier **631** that is stored in the shadow stack **630** equals ID(ENV1). In this case, the flow continues with the execution of a jump to second software environment command.

[0098] Otherwise, the shadow stack **630** will not store ID(ENV1) and an attack is detected.

[0099] FIG. **7** illustrates an example of a second software environment **640**, and a shadow stack **630**.

[0100] The second software environment **640** includes a second entry region **641** and one or more other parts **642** that include a second gadget **651**, a flow change command such as return **652** (return to the first software environment), and a check control flow integrity command **654**.

[0101] When the flow successfully jumped from the first software environment to the second entry region **641** of the second software environment, it executed an update shadow stack with ID(ENV2) command **643**, and the shadow stack **630** stores ID(ENV2) **632**. The shadow stack **630** also stores, as a return environment identifier ID(ENV1) **631**.

[0102] In this case, the execution of check control flow integrity command **654** will indicate that the return command may be executed, and the flow will use return environment identifier ID(ENV1) **631** for returning to the first software environment.

[0103] If second gadget **651** was reached without passing through the second entry region **641**, then the check control integrity command **654** will find that ID(ENV2) differs from the shadow environment identifier (for example ID(ENV1)) and an attack is detected.

[0104] Any machine language program mentioned above may be executed by the processing unit **110** of FIGS. **2**A-**2**D, or by any processor that may include one or more processing circuits. A processing circuit may include one or more field programmable gate arrays (FPGAs), one of more graphical processing units (GPUs), one of more general purpose units, one of more central processing units (CPUs), one or more hardware accelerators, one or more integrated circuits, and the like.

[0105] Any of method describing steps may include more steps than those illustrated in the figure, only the steps illustrated in the figure or substantially only the steps illustrated in the figure. The same applies to components of a device, processor or system and to instructions stored in any non-transitory computer readable storage medium.

[0106] The invention may also be implemented in a computer program for running on a computer system, at least including code portions for performing steps of a method according to the invention when run on a programmable apparatus, such as a computer system or enabling a programmable apparatus to perform functions of a device or system according to the invention. The computer program may cause the storage system to allocate disk drives to disk drive groups.

[0107] A computer program is a list of instructions such as a particular application program and/or an operating system. The computer program may for instance include one or more of: a subroutine, a function, a procedure, an object method, an object implementation, an executable application, an applet, a servlet, a source code, an object code, a shared library/dynamic load library and/or other sequence of instructions designed for execution on a computer system.

[0108] The computer program may be stored internally on a non-transitory computer readable medium. All or some of the computer program may be provided on computer readable media permanently, removably or remotely coupled to an information processing system. The computer readable media may include, for example and without limitation, any number of the following: magnetic storage media including disk and tape storage media; optical storage media such as compact disk media (e.g., CD-ROM, CD-R, etc.) and digital video disk storage media; nonvolatile memory storage media including semiconductor-based memory units such as flash memory, EEPROM, EPROM, ROM; ferromagnetic digital memories; MRAM; volatile storage media including registers, buffers or caches, main memory, RAM, etc.

[0109] A computer process typically includes an executing (running) program or portion of a program, current program values and state information, and the resources used by the operating system to manage the execution of the process. An operating system (OS) is the software that manages the sharing of the resources of a computer and provides programmers with an interface used to access those resources. An operating system processes system data and user input, and responds by allocating and managing tasks and internal system resources as a service to users and programs of the system.

[0110] The computer system may for instance include at least one processing unit or processing circuitry, associated memory and a number of input/output (I/O) devices. When executing the computer program, the computer system processes information according to the computer program and produces resultant output information via I/O devices.

Examples

[0111] Example 1 is a system comprising a processing circuit that is configured to: detect that a flow reached a flow change command or is about to reach the flow change command, wherein the flow change command belongs to a current software environment, wherein the current software environment is identified by a current environment identifier; retrieve a shadow environment identifier that is a last environment identifier stored in a shadow stack, wherein the shadow environment identifier identifies a software environment having an entry region that was a last entry region

accessed by the flow, wherein the entry region comprises a shadow stack update instruction that was executed by the flow; compare the shadow environment identifier to the current environment identifier; and detect a potential attack when the shadow environment identifier differs from the current environment identifier.

[0112] In Example 2, the subject matter of Example 1 includes, wherein the current environment identifier is stored in proximity to the flow change command.

[0113] In Example 3, the subject matter of Examples 1-2 includes, wherein the detecting of the potential attack is followed by stopping an execution of the flow.

[0114] In Example 4, the subject matter of Examples 1-3 includes, wherein the entry region was amended by a complier to include the shadow stack update instruction.

[0115] In Example 5, the subject matter of Examples 1-4 includes, amending, by a complier, the entry region to include the shadow stack update instruction.

[0116] In Example 6, the subject matter of Examples 1-5 includes, executing the flow change command when the shadow environment identifier equals the current environment identifier.

[0117] In Example 7, the subject matter of Example 6 includes, wherein the flow change command is a return command; and wherein the executing of the return command comprises: (i) retrieving, from the shadow stack, a return environment identifier that identifies a return software environment; and (ii) jumping to the return software environment.

[0118] In Example 8, the subject matter of Examples 6-7 includes, wherein the flow change command is a jump subroutine command for jumping to a subroutine; and wherein the executing of the jump subroutine command comprises: jumping to the subroutine and executing a shadow stack update included in the entry region of the subroutine; wherein the executing of the shadow stack update comprises storing in the shadow stack a shadow environment identifier that identifies the subroutine.

[0119] In Example 9, the subject matter of Examples 6-8 includes, wherein the flow change command is a jump indirect command for jumping to an address that is stored in a memory element; wherein the address belongs to the current software environment.

[0120] In Example 10, the subject matter of Examples 1-9 includes, wherein the flow change command is a jump subroutine command.

[0121] In Example 11, the subject matter of Examples 1-10 includes, wherein the flow change command is a return command.

[0122] In Example 12, the subject matter of Examples 1-11 includes, wherein the flow change command is jump indirect command.

[0123] Example 13 is a method for evaluating flow control integrity, the method comprises: detecting that a flow reached a flow change command or is about to reach the flow change command, wherein the flow change command belongs to a current software environment, wherein the current software environment is identified by a current environment identifier; retrieving a shadow environment identifier that is a last environment identifier stored in a shadow stack, wherein the shadow environment identifier identifies a software environment having an entry region that was a last entry region accessed by the flow, wherein the entry region comprises a shadow stack update instruction that was executed by the flow; comparing the shadow environment identifier to the current environment identifier; and detecting a potential attack when the shadow environment identifier differs from the current environment identifier.

[0124] In Example 14, the subject matter of Example 13 includes, wherein the current environment identifier is stored in proximity to the flow change command.

[0125] In Example 15, the subject matter of Examples 13-14 includes, wherein the detecting of the potential attack is followed by stopping an execution of the flow.

[0126] In Example 16, the subject matter of Examples 13-15 includes, wherein the entry region was amended by a complier to include the shadow stack update instruction.

[0127] In Example 17, the subject matter of Examples 13-16 includes, amending, by a complier, the entry region to include the shadow stack update instruction.

[0128] In Example 18, the subject matter of Examples 13-17 includes, executing the flow change command when the shadow environment identifier equals the current environment identifier.

[0129] In Example 19, the subject matter of Example 18 includes, wherein the flow change command is a return command; and wherein the executing of the return command comprises: (i) retrieving, from the shadow stack, a return environment identifier that identifies a return software environment; and (ii) jumping to the return software environment.

[0130] In Example 20, the subject matter of Examples 18-19 includes, wherein the flow change command is a jump subroutine command for jumping to a subroutine; and wherein the executing of the jump subroutine command comprises: jumping to the subroutine and executing a shadow stack update included in the entry region of the subroutine; wherein the executing of the shadow stack update comprises storing in the shadow stack a shadow environment identifier that identifies the subroutine.

[0131] In Example 21, the subject matter of Examples 18-20 includes, wherein the flow change command is a jump indirect command for jumping to an address that is stored in a memory element; wherein the address belongs to the current software environment.

[0132] In Example 22, the subject matter of Examples 13-21 includes, wherein the flow change command is a jump subroutine command.

[0133] In Example 23, the subject matter of Examples 13-22 includes, wherein the flow change command is a return command.

[0134] In Example 24, the subject matter of Examples 13-23 includes, wherein the flow change command is jump indirect command.

[0135] Example 25 is a non-transitory computer readable medium that stores instructions, which when executed on a processing circuit, causes the processing circuit to perform operations comprising: detecting that a flow reached a flow change command or is about to reach the flow change command, wherein the flow change command belongs to a current software environment, wherein the current software environment is identified by a current environment identifier; retrieving a shadow environment identifier that is a last environment identifier stored in a shadow stack, wherein the shadow environment identifier identifies a software environment having an entry region that was a last entry region accessed by the flow, wherein the entry region comprises a

shadow stack update instruction that was executed by the flow; comparing the shadow environment identifier to the current environment identifier; and detecting a potential attack when the shadow environment identifier differs from the current environment identifier.

[0136] In Example 26, the subject matter of Example 25 includes, wherein the current environment identifier is stored in proximity to the flow change command.

[0137] In Example 27, the subject matter of Examples 25-26 includes, wherein the detecting of the potential attack is followed by stopping an execution of the flow.

[0138] In Example 28, the subject matter of Examples 25-27 includes, wherein the entry region was amended by a complier to include the shadow stack update instruction.

[0139] In Example 29, the subject matter of Examples 25-28 includes, that stores instructions for amending, by a complier, the entry region to include the shadow stack update instruction.

[0140] In Example 30, the subject matter of Examples 25-29 includes, that stores instructions for executing the flow change command when the shadow environment identifier equals the current environment identifier.

[0141] In Example 31, the subject matter of Example 30 includes, wherein the flow change command is a return command; and wherein the executing of the return command comprises: (i) retrieving, from the shadow stack, a return environment identifier that identifies a return software environment; and (ii) jumping to the return software environment.

[0142] In Example 32, the subject matter of Examples 30-31 includes, wherein the flow change command is a jump subroutine command for jumping to a subroutine; and wherein the executing of the jump subroutine command comprises: jumping to the subroutine and executing a shadow stack update included in the entry region of the subroutine; wherein the executing of the shadow stack update comprises storing in the shadow stack a shadow environment identifier that identifies the subroutine.

[0143] In Example 33, the subject matter of Examples 30-32 includes, wherein the flow change command is a jump indirect command for jumping to an address that is stored in a memory element; wherein the address belongs to the current software environment.

[0144] In Example 34, the subject matter of Examples 25-33 includes, wherein the flow change command is a jump subroutine command.

[0145] In Example 35, the subject matter of Examples 25-34 includes, wherein the flow change command is a return command.

[0146] In Example 36, the subject matter of Examples 25-35 includes, wherein the flow change command is jump indirect command.

[0147] Example 37 is at least one machine-readable medium including instructions that, when executed by processing circuitry, cause the processing circuitry to perform operations to implement of any of Examples 1-36.

[0148] Example 38 is an apparatus comprising means to implement of any of Examples 1-36.

[0149] Example 39 is a system to implement of any of Examples 1-36.

[0150] Example 40 is a method to implement of any of Examples 1-36.

[0151] In the foregoing specification, the invention has been described with reference to specific examples of embodiments of the invention. It will, however, be evident that various modifications and changes may be made therein without departing from the broader spirit and scope of the invention as set forth in the appended claims.

[0152] Moreover, the terms "front," "back," "top," "bottom," "over," "under" and the like in the description and in the claims, if any, are used for descriptive purposes and not necessarily for describing permanent relative positions. It is understood that the terms so used are interchangeable under appropriate circumstances such that the embodiments of the invention described herein are, for example, capable of operation in other orientations than those illustrated or otherwise described herein.

[0153] The connections as discussed herein may be any type of connection suitable to transfer signals from or to the respective nodes, units or devices, for example via intermediate devices. Accordingly, unless implied or stated otherwise, the connections may for example be direct connections or indirect connections. The connections may be illustrated or described in reference to being a single connection, a plurality of connections, unidirectional connections, or bidirectional connections. However, different embodiments may vary the implementation of the connections. For example, separate unidirectional connections may be used rather than bidirectional connections and vice versa. Also, plurality of connections may be replaced with a single connection that transfers multiple signals serially or in a time multiplexed manner. Likewise, single connections carrying multiple signals may be separated out into various different connections carrying subsets of these signals. Therefore, many options exist for transferring signals.

[0154] Although specific conductivity types or polarity of potentials have been described in the examples, it will be appreciated that conductivity types and polarities of potentials may be reversed.

[0155] Each signal described herein may be designed as positive or negative logic. In the case of a negative logic signal, the signal is active low where the logically true state corresponds to a logic level zero. In the case of a positive logic signal, the signal is active high where the logically true state corresponds to a logic level one. Note that any of the signals described herein may be designed as either negative or positive logic signals. Therefore, in alternate embodiments, those signals described as positive logic signals may be implemented as negative logic signals, and those signals described as negative logic signals may be implemented as positive logic signals.

[0156] Furthermore, the terms "assert" or "set" and "negate" (or "deassert" or "clear") are used herein when referring to the rendering of a signal, status bit, or similar apparatus into its logically true or logically false state, respectively. If the logically true state is a logic level one, the logically false state is a logic level zero. And if the logically true state is a logic level zero, the logically false state is a logic level one.

[0157] Those skilled in the art will recognize that the boundaries between logic blocks are merely illustrative and that alternative embodiments may merge logic blocks or circuit elements or impose an alternate decomposition of functionality upon various logic blocks or circuit elements. Thus, it is to be understood that the architectures depicted herein are merely exemplary, and that in fact many other architectures may be implemented which achieve the same functionality.

[0158] Any arrangement of components to achieve the same functionality is effectively "associated" such that the desired functionality is achieved. Hence, any two components herein combined to achieve a particular functionality may be seen as "associated with" each other such that the desired functionality is achieved, irrespective of architectures or intermedial components. Likewise, any two components so associated can also be viewed as being "operably connected," or "operably coupled," to each other to achieve the desired functionality.

[0159] Furthermore, those skilled in the art will recognize that boundaries between the above described operations merely illustrative. The multiple operations may be combined into a single operation, a single operation may be distributed in additional operations and operations may be executed at least partially overlapping in time. Moreover, alternative embodiments may include multiple instances of a particular operation, and the order of operations may be altered in various other embodiments.

[0160] Also for example, in one embodiment, the illustrated examples may be implemented as circuitry located on a single integrated circuit or within a same device. Alternatively, the examples may be implemented as any number of separate integrated circuits or separate devices interconnected with each other in a suitable manner.

[0161] Also for example, the examples, or portions thereof, may implemented as soft or code representations of physical circuitry or of logical representations convertible into physical circuitry, such as in a hardware description language of any appropriate type.

[0162] Also, the invention is not limited to physical devices or units implemented in non-programmable hardware but can also be applied in programmable devices or units able to perform the desired device functions by operating in accordance with suitable program code, such as mainframes, minicomputers, servers, workstations, personal computers, notepads, personal digital assistants, electronic games, automotive and other embedded systems, cell phones and various other wireless devices, commonly denoted in this application as 'computer systems'.

[0163] However, other modifications, variations and alternatives are also possible. The specifications and drawings are, accordingly, to be regarded in an illustrative rather than in a restrictive sense.

[0164] In the claims, any reference signs placed between parentheses shall not be construed as limiting the claim. The word 'comprising' does not exclude the presence of other elements or steps then those listed in a claim. Furthermore, the terms "a" or "an," as used herein, are defined as one or more than one. Also, the use of introductory phrases such as "at least one" and "one or more" in the claims should not be construed to imply that the introduction of another claim element by the indefinite articles "a" or "an" limits any particular claim containing such introduced claim element to inventions containing only one such element, even when the same claim includes the introductory phrases "one or more" or "at least one" and indefinite articles such as "a" or "an." The same holds true for the use of definite articles. Unless stated otherwise, terms such as "first" and "second" are used to arbitrarily distinguish between the elements such terms describe. Thus, these terms are not necessarily intended to indicate temporal or other prioritization of such elements. The mere fact that certain measures are recited in mutually different claims does not indicate that a combination of these measures cannot be used to advantage.

[0165] While certain features of the invention have been illustrated and described herein, many modifications, substitutions, changes, and equivalents will now occur to those of ordinary skill in the art. It is, therefore, to be understood that the appended claims are intended to cover all such modifications and changes as fall within the true spirit of the invention.

1.-**38**. (canceled)

39. A system comprising a processing circuit that is configured to:

detect that a flow reached a flow change command or is about to reach the flow change command, wherein the flow change command belongs to a current software environment, wherein the current software environment is identified by a current environment identifier;

retrieve a shadow environment identifier that is a last environment identifier stored in a shadow stack, wherein the shadow environment identifier identifies a software environment having an entry region that was a last entry region accessed by the flow, wherein the entry region comprises a shadow stack update instruction that was executed by the flow;

compare the shadow environment identifier to the current environment identifier; and

detect a potential attack when the shadow environment identifier differs from the current environment identifier.

40. The system according to claim **39**, wherein to detect that the flow is about to reach the flow change command, the processing circuit is configured to search the flow change command in proximity to an entry pointed by a program counter.

41. The system according to claim **39**, wherein the detecting of the potential attack is followed by stopping an execution of the flow.

42. The system according to claim **39**, wherein the processing circuitry is configured to amend, by a complier, the entry region to include the shadow stack update instruction.

43. The system according to claim **39**, wherein the entry region was amended, by a complier, to include the shadow stack update instruction.

44. The system according to claim **39**, wherein the processing circuit is configured to execute the flow change command when the shadow environment identifier equals the current environment identifier.

45. The system according to claim **44**, wherein the flow change command is a return command; and wherein the executing of the return command comprises:

retrieving, from the shadow stack, a return environment identifier that identifies a return software environment; and jumping to the return software environment.

46. The system according to claim **44**, wherein the flow change command is a jump subroutine command for jumping to a subroutine; and wherein the executing of the jump subroutine command comprises: jumping to the subroutine and executing a shadow stack update included in the entry region of the subroutine; wherein the executing of the shadow stack update comprises storing in the shadow stack a shadow environment identifier that identifies the subroutine.

47. The system according to claim **44**, wherein the flow change command is a jump indirect command for jumping

to an address that is stored in a memory element; wherein the address belongs to the current software environment.

48. The system according to claim **39**, wherein the flow change command is a jump subroutine command.

49. The system according to claim **39**, wherein the flow change command is a return command.

50. The system according to claim **39**, wherein the flow change command is jump indirect command.

51. The system according to claim **39**, wherein the current software environment comprises a current software environment entry region that comprises a shadow stack update instruction for updating the shadow stack with the current environment identifier.

52. The system according to claim **39**, wherein the processing circuit is configured to detect the potential attack before executing the flow change command.

53. The system according to claim **52**, wherein the processing circuit is configured to respond to a detection of the potential attack before executing the flow change command.

54. The system according to claim **53**, wherein the processing circuit is configured to respond by preforming at least one of: generating an alert, stopping the execution of the flow, or rebooting a processor.

55. A method for evaluating flow control integrity, comprising:

detecting that a flow reached a flow change command or is about to reach the flow change command, wherein the flow change command belongs to a current software environment, wherein the current software environment is identified by a current environment identifier;

retrieving a shadow environment identifier that is a last environment identifier stored in a shadow stack, wherein the shadow environment identifier identifies a software environment having an entry region that was a last entry region accessed by the flow, wherein the entry region comprises a shadow stack update instruction that was executed by the flow;

comparing the shadow environment identifier to the current environment identifier; and

detecting a potential attack when the shadow environment identifier differs from the current environment identifier.

56. The method according to claim **55**, wherein the detecting that the flow is about to reach the flow change command comprises searching the flow change command in proximity to an entry pointed by a program counter.

57. The method according to claim **55**, wherein the detecting of the potential attack is followed by stopping an execution of the flow.

58. The method according to claim **55**, wherein the entry region was amended by a complier to include the shadow stack update instruction.

59. The method according to claim **55**, comprising amending, by a complier, the entry region to include the shadow stack update instruction.

60. The method according to claim **55**, comprising executing the flow change command when the shadow environment identifier equals the current environment identifier.

61. The method according to claim **60**, wherein the flow change command is a return command; and wherein the executing of the return command comprises: retrieving, from the shadow stack, a return environment identifier that identifies a return software environment; and jumping to the return software environment.

62. The method according to claim **60**, wherein the flow change command is a jump subroutine command for jumping to a subroutine; and wherein the executing of the jump subroutine command comprises: jumping to the subroutine and executing a shadow stack update included in the entry region of the subroutine; wherein the executing of the shadow stack update comprises storing in the shadow stack a shadow environment identifier that identifies the subroutine.

63. The method according to claim **60**, wherein the flow change command is a jump indirect command for jumping to an address that is stored in a memory element; wherein the address belongs to the current software environment.

64. The method according to claim **55**, wherein the flow change command is a jump subroutine command.

65. The method according to claim **55**, wherein the flow change command is a return command.

66. The method according to claim **55**, wherein the flow change command is jump indirect command.

67. The method according to claim **55**, wherein the current software environment comprises a current software environment entry region that comprises a shadow stack update instruction for updating the shadow stack with the current environment identifier.

68. The method according to claim **55**, wherein the detecting of the potential attack occurs before executing the flow change command.

69. The method according to claim **68**, comprising responding to the detecting of the potential attack before executing the flow change command.

70. The method according to claim **69**, wherein the responding comprises at least one of: generating an alert, stopping the execution of the flow, or rebooting a processor.

71. A non-transitory computer readable medium that stores instructions, which when executed on a processing circuit, causes the processing circuit to perform operations comprising:

detecting that a flow reached a flow change command or is about to reach the flow change command, wherein the flow change command belongs to a current software environment, wherein the current software environment is identified by a current environment identifier;

retrieving a shadow environment identifier that is a last environment identifier stored in a shadow stack, wherein the shadow environment identifier identifies a software environment having an entry region that was a last entry region accessed by the flow, wherein the entry region comprises a shadow stack update instruction that was executed by the flow;

comparing the shadow environment identifier to the current environment identifier; and

detecting a potential attack when the shadow environment identifier differs from the current environment identifier.

* * * * *